PA4

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/605,794	06/27/2000	Osman Abdoul Ismael	P2651C	6721

32658 7590 02/23/2004

HOGAN & HARTSON LLP
ONE TABOR CENTER, SUITE 1500
1200 SEVENTEEN ST.
DENVER, CO 80202

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2126	25

DATE MAILED: 02/23/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/605,794	ISMAEL ET AL.	
	Examiner	Art Unit	
	Lewis A. Bullock, Jr.	2126	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12 December 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-18* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *24*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 1, 2, 5, 7, 8 and 15-18 are rejected under 35 U.S.C. 102(b) as being anticipated by "A Distributed Object Model for the Java System" by WOLLRATH.

As to claim 1, WOLLRATH teaches a method of managing from a client station (client virtual machine) a target object (remote object) at a remote station (server virtual machine) via a telecommunications network (pg. 223, 2nd column, "However, a caller may receive, from a remote object in a different virtual machine, a remote reference to the object whose implementing is in the same virtual machine. In this case, the client (the caller) has a reference to a stub for the remote object..."; pg. 221, 1st column, "In our model, a remote object is one whose methods can be accessed from another address space, potentially on a different machine."), the method comprising the steps of: generating a client object (stub) forming a representation of the target object (remote object) (pg. 228, 1st column, "In remote procedure call systems, client-side stub code must be generated and linked into a client before a remote procedure call can be done."), which client object is configured to extract methods of the target object which are remotely accessible (remote interfaces) and support manipulation of properties of the target object (via remote procedure calls / remote method invocations on the remote

interfaces of the stub), the client object (stub) being further configured to implement the remotely accessible methods (remote interfaces) (pg. 228, 2nd column, "In this scheme, client-side stub code is generated from the remote object implementation class, and therefore supports the same set of remote interfaces as supported by the remote implementation. Such stub code resides on the server's host…and can be downloaded to the client on demand…or could be generated on the client-side from the list of remote interfaces supported by the remote implementation."; pg. 222, 1st column, "The default constructor for RemoteServer takes care of making an implementation object remotely accessible to clients by exporting the remote object implementation to the RMI runtime."); registering the target object (remote object) and a network adapter (channel) for a network protocol with a framework (transport layer) at the remote station; associating the client object (stub) with a network adaptor (channel) for the network protocol at the client machine (pg. 227, 1st and 2nd columns, "In general, the transport of the RMI system is responsible for: setting up connections to remote address spaces...Thus, client and server transports can negotiate to find a common transport between them."; pg. 226, 1st column, "The transport is responsible for connection set-up with remote locations and connection management,…and then up through the transport, remote reference layer and stub on the client side."); and enabling a client application (client application) to access the methods which support remote manipulation (remote interfaces) by instantiating the client object (stub) (pg. 226-228; fig. 3 pg. 225).

As to claim 18, refer to claim 1 for rejection. However, claim 18 further details the client station is a first virtual machine and the remote station is a second virtual machine and that the target object includes features that can be manipulated programmatically by processes in the second virtual machine. WOLLRATH teaches the one of the goals for supporting distributed objects in Java is supporting seamless remote invocation between Java objects in different virtual machines (pg. 219, 2nd column, final paragraph) and that a remote object is one whose methods can be accessed from another address space wherein remote method invocation (or RMI) is the action of invoking a method (of a remote interface) on a remote object (pg. 221, 1st column, Distributed Object Model). Therefore, WOLLRATH teaches the client station is a first virtual machine and the remote station is a second virtual machine since the goal is to supported remote invocation between Java objects in different virtual machines and the target object includes features that can be manipulated programmatically by processes in the second virtual machine since the target object contains methods that are accessed and invoked from another address space through a RMI.

As to claim 7, WOLLRATH teaches a remote access support mechanism (RMI system) at a client station (client virtual machine) for accessing a target object (remote object) at a remote station (server virtual machine) via a telecommunications network (pg. 223, 2nd column, "However, a caller may receive, from a remote object in a different virtual machine, a remote reference to the object whose implementing is in the same virtual machine. In this case, the client (the caller) has a reference to a stub for the

remote object..."; pg. 221, 1st column, "In our model, a remote object is one whose methods can be accessed from another address space, potentially on a different machine."), the remote access support mechanism comprising: a client object (stub) forming a representation of the target object (remote object) (pg. 228, 1st column, "In remote procedure call systems, client-side stub code must be generated and linked into a client before a remote procedure call can be done."), which client object (stub) extracts methods of the target object (remote object) which are accessible remotely (remote interfaces) and implements management methods for accessing the remote accessible methods (via remote procedure calls / remote method invocations on the remote interfaces of the stub) (pg. 228, 2nd column, "In this scheme, client-side stub code is generated from the remote object implementation class, and therefore supports the same set of remote interfaces as supported by the remote implementation. Such stub code resides on the server's host...and can be downloaded to the client on demand...or could be generated on the client-side from the list of remote interfaces supported by the remote implementation."; pg. 222, 1st column, "The default constructor for RemoteServer takes care of making an implementation object remotely accessible to clients by exporting the remote object implementation to the RMI runtime."); and a network adaptor (channel) responsive to the client object (stub) (pg. 227, 1st and 2nd columns, "In general, the transport of the RMI system is responsible for: setting up connections to remote address spaces...Thus, client and server transports can negotiate to find a common transport between them."; pg. 226, 1st column, "The transport is responsible for connection set-up with remote locations and connection

management,...and then up through the transport, remote reference layer and stub on the client side."), wherein the client object (stub) is configured to be instantiated by a client application (client application) for enabling the client application (client application) to access and modify the target object (remote object) (pg. 226-228; fig. 3 pg. 225).

As to claim 15, WOLLRATH teaches a remote access support mechanism (RMI system) at a first machine (server virtual machine) permitting remote access and modification from a client machine (client virtual machine) to a target object (remote object) at a first machine (server virtual machine) via a telecommunications network (pg. 223, 2nd column, "However, a caller may receive, from a remote object in a different virtual machine, a remote reference to the object whose implementing is in the same virtual machine. In this case, the client (the caller) has a reference to a stub for the remote object..."; pg. 221, 1st column, "In our model, a remote object is one whose methods can be accessed from another address space, potentially on a different machine."), the remote access support mechanism comprising: at least one target object (remote object); at least one network adaptor (channel) supporting a network protocol; the at least one target object (remote object) and the at least one network adaptor (channel) being registerable with a framework (transport layer) at the first machine (server virtual machine) and the network adaptor being responsive to remote access requests (RMI requests) from the client machine (client virtual machine) in accordance with the protocol (pg. 227, 1st and 2nd columns, "In general, the transport of the RMI system is responsible for: setting up connections to remote address

spaces...Thus, client and server transports can negotiate to find a common transport between them."; pg. 226, 1st column, "The transport is responsible for connection set-up with remote locations and connection management,...and then up through the transport, remote reference layer and stub on the client side.") to extract target object methods (in order to generate a stub by dynamic stub loading from the server) and modify the target object via the framework (via RMI calls) (pg. 225-228).

As to claim 16, WOLLRATH teaches the support mechanism (RMI system) comprising a software mechanism (layers) (fig. 3 on page 225).

As to claim 17, refer to claim 15 for rejection.

As to claim 2, WOLLRATH teaches compiling the target object (remote object implementation class) to generate the client object (client-side stub code) (pg. 228, 2nd column, "In this scheme, client-side stub code is generated from the remote object implementation class, and therefore supports the same set of remote interfaces as supported by the remote implementation."), the client object (stub) comprising a target object interface identifying the remotely accessible methods (remote interfaces) of the target object (remote object) and the client object (stub) further comprising a target object stub implementing the remotely accessible methods (client object is a stub containing the remote interfaces) (pg. 228; pg. 223, "In the distributed object model,

clients interact with stub (surrogate) objects that have exactly the same set of remote interfaces defined by the remote object's class...").

As to claim 5, WOLLRATH teaches extracting target object methods (remote interfaces) by introspection (pg. 228, 2nd column, "Stub code for a remote implementation....could be generated on the client-side from the list of remote interfaces supported by the remote implementation."; pg. 222, 1st column, "The default constructor for RemoteServer takes care of making an implementation object remotely accessible to clients by exporting the remote object implementation to the RMI runtime.").

As to claim 8, refer to claim 2 for rejection.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 3 and 9 are rejected under 35 U.S.C. 103(a) as being unpatentable over "A Distributed Object Model for the Java System" by WOLLRATH in view of "Specializing Object-Oriented RPC for Functionality and Performance" by ZELESKO.

As to claim 3, WOLLRATH teaches the creation of a client object (stub) as a local representation of a target object (remote object) wherein the target objects' methods (remote interfaces) are extracted (exported) and placed in the client object (stub) (pg. 228, 2nd column, "In this scheme, client-side stub code is generated from the remote object implementation class, and therefore supports the same set of remote interfaces as supported by the remote implementation. Such stub code resides on the server's host...and can be downloaded to the client on demand...or could be generated on the client-side from the list of remote interfaces supported by the remote implementation."; pg. 222, 1st column, "The default constructor for RemoteServer takes care of making an implementation object remotely accessible to clients by exporting the remote object implementation to the RMI runtime."). However, WOLLRATH does not teach the selectively replacing the target object stub.

ZELESKO teaches the creation of a client object (local proxy or stub) as a local representation of a target object (object) wherein the target object's methods are placed in the client object ("The proxy and the concrete implementation of the object each inherit the exported interface...") (pg. 176, 1st column) and selectively replacing the target object stub (client stub / server stub) for dynamically modifying the behavior of the client application (client) at runtime (pg. 176, 2nd column, "...(3) extend the functionality of client and server stubs."; pg. 184, "Proxy specialization refers to extending the functionality of client stubs beyond simple call forwarding...In such a case the proxy can respond locally to queries of the attribute, eliminating the need to forward the call over the network to the remote object .."). Therefore, it would be obvious to one skilled in the

art at the time of the invention to combine the teachings of WOLLRATH with the teachings of ZELESKO in order to define substantially different functionality for the object corresponding to a proxy (pg. 185).

As to claim 9, refer to claim 3 for rejection.

5. Claim 4 is rejected under 35 U.S.C. 103(a) as being unpatentable over "A Distributed Object Model for the Java System" by WOLLRATH in view of "IBM VisualAge for Java, Getting Started for OS/2* and for Windows**, Version 1.0" by IBM.

As to claim 4, WOLLRATH teaches the creation of a client object (stub) as a local representation of a target object (remote object) wherein the target objects' methods (remote interfaces) are extracted (exported) and placed in the client object (stub) (pg. 228, 2nd column, "In this scheme, client-side stub code is generated from the remote object implementation class, and therefore supports the same set of remote interfaces as supported by the remote implementation. Such stub code resides on the server's host...and can be downloaded to the client on demand...or could be generated on the client-side from the list of remote interfaces supported by the remote implementation."; pg. 222, 1st column, "The default constructor for RemoteServer takes care of making an implementation object remotely accessible to clients by exporting the remote object implementation to the RMI runtime."). However, WOLLRATH does not teach the client object and the target object are beans.

IBM teaches the creation of a client object (client proxy bean) as a local representation of a target object (server bean) wherein the client object (client-side server proxy which is a non-visual bean) and the target object (server bean) are beans, each of which comprises a set of properties, a set of methods for performing actions, and support for events and for introspection (via RMI Access Builder) (pg. 132-137). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of WOLLRATH with IBM in order to facilitate remote access to your beans in a client-server environment (pg. 132).

6. Claim 6 is rejected under 35 U.S.C. 103(a) as being unpatentable over "A Distributed Object Model for the Java System" by WOLLRATH in view of HOLLBERG (EP 0727739 A1).

As to claim 6, WOLLRATH teaches the creation of a client object (stub) as a local representation of a target object (remote object) wherein the target objects' methods (remote interfaces) are extracted (exported) and placed in the client object (stub) (pg. 228, 2nd column, "In this scheme, client-side stub code is generated from the remote object implementation class, and therefore supports the same set of remote interfaces as supported by the remote implementation. Such stub code resides on the server's host...and can be downloaded to the client on demand...or could be generated on the client-side from the list of remote interfaces supported by the remote implementation."; pg. 222, 1st column, "The default constructor for RemoteServer takes care of making an implementation object remotely accessible to clients by exporting the remote object

implementation to the RMI runtime."). However, WOLLRATH does not teach the target object is a managed object.

HOLLBERG teaches the creation of a client object (proxy managed object) as a local representation of a target object (remote managed object) wherein the target object is a managed object (remote managed object) and the client application is a network management application (application access managed objects through proxy managed objects) (pg. 4, lines 13-33; pg. 2-3). Therefore, it would be obvious to one skilled in the art to combine the teachings of WOLLRATH with the teachings of HOLLBERG in order to separate the application from communication interfaces and technologies through a generic communication class (pg. 6, lines 5-35).

7. Claims 10-12 are rejected under 35 U.S.C. 103(a) as being unpatentable over WOLLRATH in view of ZELESKO as applied to claim 9 above, and further in view of "IBM VisualAge for Java, Getting Started for OS/2 and for Windows**, Version 1.0" by IBM.

As to claims 10 and 11, WOLLRATH teaches the creation of a client object (stub) as a local representation of a target object (remote object) wherein the target objects' methods (remote interfaces) are extracted (exported) and placed in the client object (stub) (pg. 228, 2nd column, "In this scheme, client-side stub code is generated from the remote object implementation class, and therefore supports the same set of remote interfaces as supported by the remote implementation. Such stub code resides on the server's host...and can be downloaded to the client on demand...or could be generated

on the client-side from the list of remote interfaces supported by the remote implementation."; pg. 222, 1st column, "The default constructor for RemoteServer takes care of making an implementation object remotely accessible to clients by exporting the remote object implementation to the RMI runtime."). However, neither WOLLRATH nor ZELESKO teach the client object and the target object are beans.

IBM teaches the creation of a client object (client proxy bean) as a local representation of a target object (server bean) wherein the client object (client-side server proxy which is a non-visual bean) and the target object (server bean) are beans, each of which comprises a set of properties, a set of methods for performing actions, and support for events and for introspection (via RMI Access Builder) (pg. 132-137). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of WOLLRATH with ZELESKO and IBM in order to facilitate remote access to your beans in a client-server environment (pg. 132).

As to claim 12, IBM teaches a compiler (RMI compiler) for extracting target object methods (server object inherited methods) by introspection for generating the client object (client-side server proxy which is a non-visual bean) (pg. 132-137).

8. Claims 13 and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over WOLLRATH in view of ZOLESKO and IBM as applied to claim 11 above, and further in view of HOLLBERG (EP 0727739 A1).

As to claim 13, WOLLRATH teaches the creation of a client object (stub) as a local representation of a target object (remote object) wherein the target objects' methods (remote interfaces) are extracted (exported) and placed in the client object (stub) (pg. 228, 2nd column, "In this scheme, client-side stub code is generated from the remote object implementation class, and therefore supports the same set of remote interfaces as supported by the remote implementation. Such stub code resides on the server's host...and can be downloaded to the client on demand...or could be generated on the client-side from the list of remote interfaces supported by the remote implementation."; pg. 222, 1st column, "The default constructor for RemoteServer takes care of making an implementation object remotely accessible to clients by exporting the remote object implementation to the RMI runtime."). However, none of the combination of references teach the target object is a managed object.

HOLLBERG teaches the creation of a client object (proxy managed object) as a local representation of a target object (remote managed object) wherein the target object is a managed object (remote managed object) and the client application is a network management application (application access managed objects through proxy managed objects) (pg. 4, lines 13-33; pg. 2-3). Therefore, it would be obvious to one skilled in the art to combine the teachings of WOLLRATH with the teachings of ZOLESKO, IBM, and HOLLBERG in order to separate the application from communication interfaces and technologies through a generic communication class (pg. 6, lines 5-35).

As to claim 14, WOLLRATH teaches the support mechanism (RMI system) comprising a software mechanism (layers) (fig. 3 on page 225).

Response to Arguments

9. Applicant's arguments with respect to claims 1-18 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (703) 305-0439. The examiner can normally be reached on Monday-Friday, 8:30 am - 5:00 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (703) 305-9678. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only.

For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Lewis A. Bullock Jr.

lab